## Android Applications

Android applications are software programs designed to run on the Android operating system, which powers billions of smartphones and tablets worldwide. These apps offer a wide range of functionalities, from communication and entertainment to productivity and education.

### Main building blocks of an android app

The main building blocks of an Android application are its components, which work together to provide the functionality and user experience we see in the final app.

Activities**:** Activities are the fundamental building blocks that represent individual screens within your app. Each activity has two key aspects:

* **Layout:** Defines the visual structure of the screen using XML files. This layout includes UI elements like buttons, text views, images, and more.
* **Code:** Written in Java or Kotlin, this code handles user interactions with the UI elements, performs tasks, and manages the activity's lifecycle (creation, destruction, etc.).

Services**:** Unlike activities, services are background processes that run in the background. They do not have a user interface and are designed for long-running operations or tasks that do not require direct user interaction. Examples include:

* Playing music in the background
* Downloading data from the internet
* Performing complex calculations

Broadcast Receivers**:** These components respond to system events or broadcasts sent by other apps. They act like listeners, waiting for specific events to occur and then triggering predefined actions within your app. Examples include:

* Reacting to changes in network connectivity (Wi-Fi or mobile data)
* Responding to system boot completion
* Receiving SMS messages (depending on permissions)

Content Providers**:** Act as a bridge between your app and other apps, enabling them to share data securely. They provide a standardized way to access and modify data managed by your app or other apps on the device. Examples include:

* Sharing contact information with another app
* Accessing data from a calendar app

Intents**:** Intents are messages used for communication between components within your app or even between your app and other apps. They act like messengers, carrying information about what action needs to be performed and any relevant data. Intents are crucial for launching activities, starting services, and sending broadcasts.

Resources**:** Resources are essential building blocks that define the look and feel of your app. They include various elements stored in separate directories within your project structure:

* **Layouts (XML):** Define the UI structure of your activities.
* **Drawables (Images):** Images used within your app (icons, backgrounds, etc.).
* **Strings (Text):** Text displayed throughout the app (labels, messages, etc.).
* **Styles:** Define visual properties like text color, font size, etc., for consistent UI elements.

Activity Manager**:** Activity Manager is a crucial system service that manages the lifecycle of all activities within your application. It ensures that activities appear and disappear at the right time, maintaining overall system performance and resource allocation. Responsibilities of activity manager.

* **Activity Lifecycle Management:**
  + The Activity Manager controls the lifecycle of activities, including:
    - Creating and launching new activities when requested by your app or the system.
    - Pausing and stopping activities when they are no longer in the foreground or need to free up resources.
    - Destroying activities when they are no longer needed.
* **Task Management:**
  + An Android "task" represents a group of related activities. The Activity Manager keeps track of the tasks running on the device and handles task switching, bringing activities from different tasks to the foreground or background as needed.
* **Back Stack Management:**
  + The Activity Manager maintains the back stack, which is a chronological history of activities launched by the user. This enables back button functionality, allowing users to navigate back to previous activities.
* **Security:**
  + The Activity Manager enforces security measures to ensure that only authorized apps can launch activities and access system resources.

**Key Characteristics:**

* **Application-wide access:** The Application Context remains available throughout your application's lifecycle, even when activities are paused or destroyed.
* **Singleton instance:** There is only one instance of the Application Context for your entire application.
* **Access to resources:** Provides access to application-level resources like shared preferences, databases, and assets stored within your project.
* **System-level operations:** Enables launching system-wide services or broadcasts that are not directly tied to a specific activity.

### Android Manifest File

The AndroidManifest.xml file is a fundamental element in every Android application. It acts as the central configuration file, providing essential information about your app to the Android system.  
**Location:** Located at the root directory of your Android project, typically named AndroidManifest.xml.

**Structure:** The file is written in XML format and has a defined structure with specific tags and attributes.

**Importance of AndroidManifest.xml:**

* The Android system relies on the information in this file to launch your app, display it correctly, and manage its interactions with the system and other apps.
* Any errors or missing information in the manifest can lead to app installation failures or unexpected behavior.

### Gradle

Gradle is an open-source build automation system widely used in Android development and other domains. It streamlines the app development process by automating tasks like:

* **Managing dependencies:** Gradle helps you manage the external libraries your project relies on. You specify these libraries and their versions in build scripts, and Gradle downloads and includes them in your project.
* **Building your app:** Gradle defines the steps involved in building your app, including compiling your code, packaging resources, and generating the final APK (Android Package Kit) file ready for deployment.
* **Automating tasks:** You can define custom tasks within Gradle scripts to automate repetitive actions during development, such as running tests or deploying your app to a device.

#### Gradle Scripts:

Gradle build instructions are written in Groovy or Kotlin DSL (domain-specific language) within files called build scripts. These scripts define configurations and instructions for Gradle to follow during the build process. Here is a breakdown of the key aspects of Gradle scripts:

* **Location:** Build scripts reside in the gradle directory within your project's root directory.
* **File Types:**
  + build.gradle: Located at the root project directory, this file defines project-wide configurations and dependencies.
  + build.gradle.kts (optional): Kotlin equivalent of the root build.gradle file.
  + Module-level build scripts (optional): You can have build.gradle files within specific modules of your project for more granular configurations.
* **Structure:** A build script typically consists of sections like:
  + **Plugins:** Define the plugins used in your project (e.g., Android Gradle plugin for building Android apps).
  + **Android block (for Android projects):** Configures aspects specific to Android app development, including build settings, dependencies, product flavors, and signing configurations.
  + **Repositories:** Specify locations where Gradle can download dependencies (e.g., Google's Maven repository).
  + **Dependencies:** Declare the external libraries your project depends on and their versions.
  + **Tasks:** Define custom tasks to automate specific actions within your build process.